EUROPEAN
SPALLATION
SOURCE

Document Type	Engineering Manual
Document number	ESS-0508480
Date	February 11, 2020
Revision	1
State	Released
Confidentiality Level	Internal
Page	1 (24)

ICS Engineering Manual

for MRF MTCA-EVR-300(U)

	Name	**Role/Title**
Owner	Javier Cereijo Garcia	Timing System Engineer
Author	Javier Cereijo Garcia	Timing System Engineer

Contents

1 Overview

At European Spallation Source (ESS), Integrated Control System (ICS) uses the Micro Research Finland (MRF) timing System[1] as its timing system of the ESS site. The consistent and up-to-date engineering manual is essential for the ESS timing system.

1.1 Scope

- This document identifies one of the MRF timing Event Receivers (EVRs) that need to be configured for an ESS subsystem that needs synchronous frequencies, trigger signals and sequences of events [1].
- This document provides the generic description of the MRF MTCA-EVR-300(U) board and its interface board IFB-300. In addition, it affords the minimal, essential, and generic information for the system configuration.
- The purpose of this document is to describe the engineering procedure and troubleshooting about how the MRF MTCA-EVR-300(U) board will be integrated in cooperation with the new ESS EPICS Environment (E3).
- This document attempts to maintain consistency with existing ESS timing system hardware as far as possible.

1.2 Target audience

This document is targeted to ICS engineers and technical stakeholders of the ESS timing system. It is assumed that the target audience has a technical background in the MRF Timing System, the Experimental Physics and Industrial Control System (EPICS) development, and a Linux environment.

2 System description

MRF Technical Reference [see 1, p45] describes EVRs as:

> *Event Receivers decode timing events and signals from an optical event stream transmitted by an Event Generator. Events and signals are received at predefined rate the event clock that is usually divided down from an accelerators main RF reference. The event receivers lock to the phase event clock of the Event Generator and are thus phase locked to the RF reference. Event Receivers convert event codes transmitted by an Event Generator to hardware*

[1] http://www.mrf.fi/

outputs. They can also generate software interrupts and store the event codes with globally distributed timestamps into FIFO memory to be read by a CPU.

ICS uses and will use the following different types of EVRs:

- MTCA-EVR-300(U)
- PCIe-EVR-300DC

The scope of this document is to cover the MTCA-EVR-300(U) board.

2.1 MTCA-EVR-300(U)

Figure 1 shows the rough physical dimensions of 181×148 mm^2 of the MTCA-EVR-300 card.

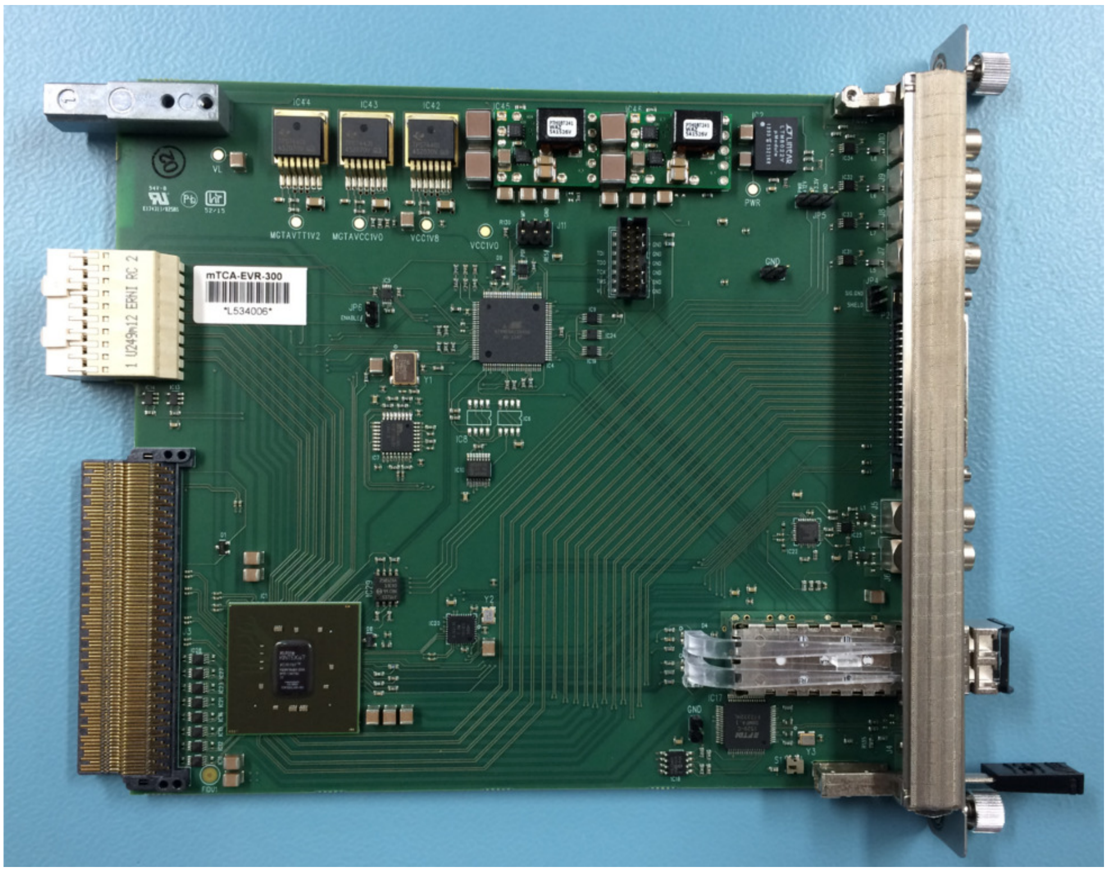

Figure 1 MRF MTCA-EVR-300 board.

The MTCA-EVR-300 has a small form-factor pluggable (SFP) transceiver as an input from an Event Master (EVM) and several outputs: 4 front panel outputs, 16 front universal outputs (through the IFB-300 extension board) and 40 rear outputs. The initial 32 rear outputs map to the Rear Transition Module (RTM) connector, the last 8 rear outputs map to the Micro Telecommunications Computing Architecture (MTCA) backplane. The 16 front universal outputs are implemented through a micro-SCSI type connector for an interface board IFB-300. The IFB-300 has eight Universal Input/Output (I/O) slots, each of them accommodating 2 outputs/inputs, shown in Figure 2. With different type of MRF Universal I/O modules, each slot can be used as an unique trigger or event signal source.

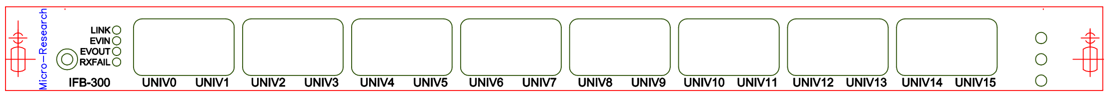

Figure 2 MRF Interface Board IFB-300 Front Panel [1].

The MTCA-EVR-300U is identical to the MTCA-EVR-300 except for the fact that it replaces the micro-SCSI connector for 2 Universal I/O module slots.

3 System environment

Before describing the engineering procedure for an E3 integration of the MRF MTCA-EVR-300(U) board, it is mandatory to have proper system environment that consists of specific hardware and software. Here we will show the hardware and software lists and their set up in the ICS lab at ESS. The information shown in this chapter is used in the ICS lab at ESS.

3.1 Hardware

Table 1 shows the hardware list. It is possible to use a different form factor for the EVM than what is shown; for more information check the specific engineering manual. It is assumed that a working EVM system is ready.

Figure 3 shows the MTCA-EVR-300 set up in the lab. From left to right, the power supply, MTCA Carrier Hub (MCH), Central Processing Unit (CPU), MTCA-EVR-300 and Struck SIS8300 (not used in this document).

3.2 Software

Table 2 shows the software list and its environment.

Hardware	Info
MRF mTCA-EVM-300	
MRF mTCA-EVR-300(U)	
μTCA crate	Incl. PM, MCH
Concurrent Technologies AMC CPU	
Optical cables	LC, optical 850 nm
Oscilloscope	Incl. cables and LEMO connector

Table 1 Hardware List.

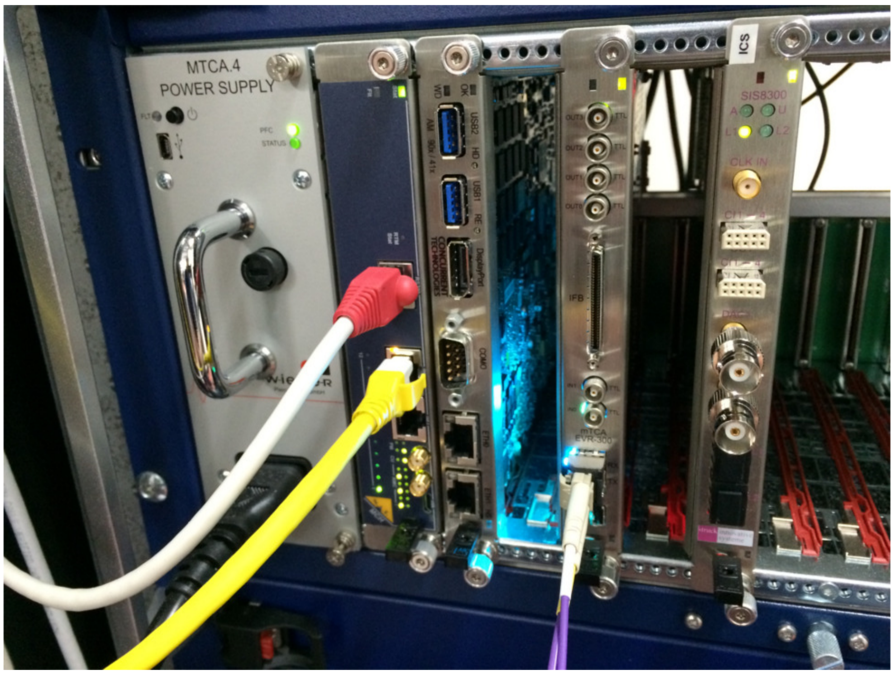

Figure 3 Hardware Setup in the ICS lab.

Item	Version Info.
CentOS Linux	`7.7.1908`
Kernel	`3.10.0-1062.9.1.el7.x86_64`
mrf kernel module	Version: `1` / srcversion `E3290AD048B5B57D2EAA55E`
EPICS base	`7.0.3.1`
e3-req	`3.1.2`
mrfioc2	E3 module ver. `2.2.0-rc8`
devLib2	E3 module ver. `2.9.0`

Table 2 Software and its version information.

3.3 EVR firmware

Table 3 shows the EVR field-programmable gate array (FPGA) Firmware Version Register.

EVR FPGA Firmware Version Register	`0x180e0207`	
Board Type	EVR	0x**1**80e0207
Form Factor	mTCA.4	0x1**8**0e0207
EVR Subrelease ID	0xe	0x18**0e**0207
EVR Firmware ID	Delay Compensation Firmware	0x180e**02**07
EVR Revision ID	7	0x180e02**07**

Table 3 EVR FPGA Firmware Version Register in Reference [see 1, p66].

4 Engineering procedure

This chapter provides the minimal information to configure the EVR board properly.

4.1 System installation

Figure 3 shows the glimpse of what system might be like in a lab. **Note that the cable between the mTCA-EVR-300 and IFB-300 (not shown in the figure) should be connected or disconnected only when the system is powered down**. Please see more detailed information in [1, p54].

It is assumed that the MTCA-EVR-300(U) is installed in a crate with a CPU running CentOS 7 with E3. For more information on E3, please check [2].

4.2 mTCA-EVR-300(U) board identification

4.2.1 Fixing PCI IDs

The PCI ID list does not include by default the MRF products. It can be updated as follows:

- Cloning the sort of ESS customized PCI.IDS db:

```
iocuser@cslab-ccpu-crate07: ~$ git clone https://github.com/jeonghanlee/pciids
```

- Replace the pci.ids file:

```
iocuser@cslab-ccpu-crate07: ~$ cd pciids/
iocuser@cslab-ccpu-crate07: pciids (master)$ bash replace-pciids.bash
centos was determined.
[sudo] password for iocuser:
iocuser@cslab-ccpu-crate07: pciids (master)$
```

- Check MRF products by the vendor's id (1a3e):

```
iocuser@cslab-ccpu-crate07: pciids (master)$ lspci -nmmn | grep -E "\<(1a3e)"
0a:00.0 "Signal processing controller [1180]" "Xilinx Corporation [10ee]" "XILINX PCI
    DEVICE [7011]" "Micro-Research Finland Oy [1a3e]" "MTCA Event Receiver 300 [132c]"
```

4.2.2 Kernel module

The MTCA-EVR-300(U) needs a kernel module to work. It can be installed by simply running some commands. Do the following:

- Go to your e3-mrfioc2 sources directory:

```
iocuser@cslab-ccpu-crate07: ~$ cd e3/e3-mrfioc2
```

- Install the kernel module:

[2] https://github.com/icshwi/e3training

```
iocuser@cslab-ccpu-crate07: e3-mrfioc2 (master)$ make dkms_add
/epics/base-7.0.3.1/bin/linux-x86_64/msi -M name="mrfioc2" -M  version="2.2.0-rc8" -M
     kmod_name="mrf" /home/iocuser/e3/e3-mrfioc2/dkms/dkms_with_msi.conf.in > /home/iocuser/
     e3/e3-mrfioc2/dkms/dkms_with_msi.conf
/usr/bin/sudo -E install -d /usr/src/mrfioc2-2.2.0-rc8
[sudo] password for iocuser:
/usr/bin/sudo -E cp -r /home/iocuser/e3/e3-mrfioc2/mrfioc2/mrmShared/linux/* /usr/src/
     mrfioc2-2.2.0-rc8/
/usr/bin/sudo -E /usr/sbin/dkms add -m mrfioc2 -v 2.2.0-rc8

Creating symlink /var/lib/dkms/mrfioc2/2.2.0-rc8/source ->
                 /usr/src/mrfioc2-2.2.0-rc8

DKMS: add completed.
iocuser@cslab-ccpu-crate07: e3-mrfioc2 (master)$ make dkms_build
/usr/bin/sudo -E /usr/sbin/dkms build -m mrfioc2 -v 2.2.0-rc8

Kernel preparation unnecessary for this kernel.  Skipping...

Building module:
cleaning build area...
make -j4 KERNELRELEASE=3.10.0-1062.9.1.el7.x86_64 -C /lib/modules/3.10.0-1062.9.1.el7.
     x86_64/build M=/var/lib/dkms/mrfioc2/2.2.0-rc8/build modules...
cleaning build area...

DKMS: build completed.
iocuser@cslab-ccpu-crate07: e3-mrfioc2 (master)$ make dkms_install
/usr/bin/sudo -E /usr/sbin/dkms install -m mrfioc2 -v 2.2.0-rc8

mrf.ko.xz:
Running module version sanity check.
 - Original module
   - No original module exists within this kernel
 - Installation
   - Installing to /lib/modules/3.10.0-1062.9.1.el7.x86_64/extra/
Adding any weak-modules

depmod...

DKMS: install completed.
iocuser@cslab-ccpu-crate07: e3-mrfioc2 (master)$ make setup
KERNEL=="uio*", ATTR{name}=="mrf-pci", MODE="0666"
mrf
rmmod mrf
rmmod parport
rmmod uio
insmod /lib/modules/3.10.0-1062.9.1.el7.x86_64/kernel/drivers/parport/parport.ko.xz
insmod /lib/modules/3.10.0-1062.9.1.el7.x86_64/kernel/drivers/uio/uio.ko.xz
insmod /lib/modules/3.10.0-1062.9.1.el7.x86_64/extra/mrf.ko.xz


It is OK to see "E3/RULES_DKMS:37: recipe for target 'setup' failed"
---------------------------------------------------------------------
crw-rw-rw- 1 root root 245, 0 Feb 04 15:37 /dev/uio0
crw-rw-rw- 1 root root 245, 1 Feb 04 15:37 /dev/uio1
---------------------------------------------------------------------
```

- Check kernel module information:

```
iocuser@cslab-ccpu-crate07: e3-mrfioc2 (master)$ lsmod |grep mrf
mrf                    18137  0
```

```
parport                46395  1 mrf
uio                    19338  1 mrf
iocuser@cslab-ccpu-crate07: e3-mrfioc2 (master)$ modinfo mrf
filename:       /lib/modules/3.10.0-1062.9.1.el7.x86_64/extra/mrf.ko.xz
author:         Michael Davidsaver <mdavidsaver@gmail.com>
version:        1
license:        GPL v2
retpoline:      Y
rhelversion:    7.7
srcversion:     E3290AD048B5B57D2EAA55E
alias:          pci:v000010EEd00007011sv00001A3Esd0000232Cbc*sc*i*
alias:          pci:v000010EEd00007011sv00001A3Esd0000132Cbc*sc*i*
alias:          pci:v00001A3Ed0000152Csv00001A3Esd0000152Cbc*sc*i*
alias:          pci:v00001A3Ed0000252Csv00001A3Esd0000252Cbc*sc*i*
alias:          pci:v000010EEd00007011sv00001A3Esd0000172Cbc*sc*i*
alias:          pci:v00001204d0000EC30sv00001A3Esd0000172Cbc*sc*i*
alias:          pci:v000010B5d00009056sv00001A3Esd0000192Cbc*sc*i*
alias:          pci:v000010B5d00009030sv00001A3Esd000011E6bc*sc*i*
alias:          pci:v000010B5d00009030sv00001A3Esd000020E6bc*sc*i*
alias:          pci:v000010B5d00009030sv00001A3Esd000020DCbc*sc*i*
alias:          pci:v000010B5d00009030sv00001A3Esd000010E6bc*sc*i*
depends:        parport,uio
vermagic:       3.10.0-1062.9.1.el7.x86_64 SMP mod_unload modversions
parm:           cable:Name of JTAG parallel port cable to emulate (charp)
parm:           interfaceversion:User space interface version (int)
parm:           use_msi:Use MSI if present (default 1, yes) (uint)
```

4.2.3 PCI addressing

Each PCI device is identified by a domain, a bus, a device, and a function number in Linux. Therefore, in order to initialize the MRF MTCA-EVR-300(U) board in E3, one needs the following information: a bus number, a device number, and a function number. These numbers are the parameters of a `mrmEvrSetupPCI` function.

One can use `lspci` to find them as follows:

```
iocuser@cslab-ccpu-crate07: e3-mrfioc2 (master)$ lspci
[...]]
0a:00.0 Signal processing controller: Xilinx Corporation XILINX PCI DEVICE
[...]
iocuser@cslab-ccpu-crate07: e3-mrfioc2 (master)$ lspci -s 0a:00.0 -vv
0a:00.0 Signal processing controller: Xilinx Corporation XILINX PCI DEVICE
   Subsystem: Micro-Research Finland Oy MTCA Event Receiver 300
   Physical Slot: 11
   Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR- FastB2B-
     DisINTx+
   Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR-
     INTx-
   Latency: 0, Cache Line Size: 64 bytes
   Interrupt: pin A routed to IRQ 46
   Region 0: Memory at c0800000 (32-bit, non-prefetchable) [size=256K]
   Capabilities: <access denied>
   Kernel driver in use: mrf-pci
   Kernel modules: mrf
```

And one should identify four number as follows:

```
iocuser@cslab-ccpu-crate07: e3-mrfioc2 (master)$ lspci -s 0a:00.0 -t
-+-[0000:0a]---00.0
```

```
\-[0000:00]-
```

, where `-+-[0000:0a]---00.0` can be translated to -+-[domain:bus]---device.function. Thus in the above case, three numbers are shown in Table 4.

bus	0x0a
device	0x00
function	0x00

Table 4 MRF MTCA-EVR-300(U) Identification Numbers

4.3 EPICS IOC set up under E3

In order to start the EPICS Input/Output Controller (IOC) for the MRF MTCA-EVR-300(U) under E3, one should consider two things: 1) the EPICS database file, and 2) the EPICS start-up script. Let us set as the working directory,

```
/home/iocuser/e3/e3-mrfioc2/cmds
```

Listing 4.1 Working directory in the ICS lab.

4.3.1 EPICS database file

The database files in E3 are located under the following directory:

```
/epics/base-7.0.3.1/require/3.1.2/siteMods/mrfioc2/2.2.0-rc8/db/evr-mtca-300-ess.db
/epics/base-7.0.3.1/require/3.1.2/siteMods/mrfioc2/2.2.0-rc8/db/evr-mtca-300u-ess.db
```

4.3.2 Start-up script

Listing 4.2 shows the IOC start-up script configured for the MRF MTCA-EVR-300(U) present in our system with the Identification Numbers obtained in previous steps.

```
require mrfioc2,2.2.0-rc8

epicsEnvSet("IOC", "EMMTCAEVR300")
epicsEnvSet("DEV1", "EVR0")

epicsEnvSet("MainEvtCODE" "14")
epicsEnvSet("HeartBeatEvtCODE"   "122")
epicsEnvSet("ESSEvtClockRate"  "88.0525")

mrmEvrSetupPCI("$(DEV1)",  "0a:00.0")
dbLoadRecords("evr-mtca-300u-ess.db","EVR=$(DEV1), SYS=$(IOC), D=$(DEV1), FEVT=$(ESSEvtClockRate),
      EVNT1HZ=$(MainEvtCODE)")

var evrMrmTimeNSOverflowThreshold 100000


```

```
iocInit()
```

Listing 4.2 Start-up script `emmtcaevr300.cmd`. Line 10 should be matched to Table 4 as `mrmEvrSetupPCI($(DEV1), "bus:device.function")`.

4.3.3 EPICS IOC

All the EPICS parameters should be run from an E3 session. To start E3, type:

```
iocuser@cslab-ccpu-crate07: cmds (master)$ source /epics/base-7.0.3.1/require/3.1.2/bin/setE3Env.
    bash

Set the ESS EPICS Environment as follows:
THIS Source NAME    : setE3Env.bash
THIS Source PATH    : /epics/base-7.0.3.1/require/3.1.2/bin
EPICS_BASE          : /epics/base-7.0.3.1
EPICS_HOST_ARCH     : linux-x86_64
E3_REQUIRE_LOCATION : /epics/base-7.0.3.1/require/3.1.2
PATH                : /epics/base-7.0.3.1/require/3.1.2/bin:/epics/base-7.0.3.1/bin/linux-x86_64:/
    usr/local/bin:/usr/bin:/usr/local/sbin:/usr/sbin:/home/iocuser/.local/bin:/home/iocuser/bin
LD_LIBRARY_PATH     : /epics/base-7.0.3.1/lib/linux-x86_64:/epics/base-7.0.3.1/require/3.1.2/lib/
    linux-x86_64:/epics/base-7.0.3.1/require/3.1.2/siteLibs/linux-x86_64

Enjoy E3!
```

All the IOCs and related EPICS commands (caget, caput, camonitor, etc.) should be run from an E3 session.

Under E3, the EPICS IOC can be started with the command `iocsh.bash emmtcaevr300.cmd`. The output should look like as follows:

```
iocuser@cslab-ccpu-crate07: cmds (master)$ iocsh.bash emmtcaevr300.cmd
registerChannelProviderLocal firstTime true
#
# ------>-----> snip ----->------>
#
# Please Use Version and other environment variables
# in order to report or debug this shell
#
# The IOC is started at "2020-W06-Feb04-1644-58-CET"
#
# Version information:
# European Spallation Source ERIC : iocsh.bash (0.5.1-81d1214.PID-24674)
#
# HOSTDISPLAY=""
# WINDOWID=""
# PWD="/home/iocuser/e3/e3-mrfioc2/cmds"
# USER="iocuser"
# LOGNAME="iocuser"
# EPICS_HOST_ARCH="linux-x86_64"
# EPICS_BASE="/epics/base-7.0.3.1"
# E3_REQUIRE_NAME="require"
# E3_REQUIRE_VERSION="3.1.2"
# E3_REQUIRE_LOCATION="/epics/base-7.0.3.1/require/3.1.2"
# E3_REQUIRE_BIN="/epics/base-7.0.3.1/require/3.1.2/bin"
# E3_REQUIRE_DB="/epics/base-7.0.3.1/require/3.1.2/db"
# E3_REQUIRE_DBD="/epics/base-7.0.3.1/require/3.1.2/dbd"
# E3_REQUIRE_INC="/epics/base-7.0.3.1/require/3.1.2/include"
# E3_REQUIRE_LIB="/epics/base-7.0.3.1/require/3.1.2/lib"
# E3_SITEAPPS_PATH="/epics/base-7.0.3.1/require/3.1.2/siteApps"
# E3_SITELIBS_PATH="/epics/base-7.0.3.1/require/3.1.2/siteLibs"
# E3_SITEMODS_PATH="/epics/base-7.0.3.1/require/3.1.2/siteMods"
# EPICS_DRIVER_PATH="/epics/base-7.0.3.1/require/3.1.2/siteMods:/epics/base-7.0.3.1/require/3.1.2/
    siteApps"
# EPICS_CA_AUTO_ADDR_LIST=""
# EPICS_CA_ADDR_LIST=""
# PATH="/epics/base-7.0.3.1/require/3.1.2/bin:/epics/base-7.0.3.1/bin/linux-x86_64:/usr/local/bin
    :/usr/bin:/usr/local/sbin:/usr/sbin:/home/iocuser/.local/bin:/home/iocuser/bin"
# LD_LIBRARY_PATH="/epics/base-7.0.3.1/lib/linux-x86_64:/epics/base-7.0.3.1/require/3.1.2/lib/
    linux-x86_64:/epics/base-7.0.3.1/require/3.1.2/siteLibs/linux-x86_64"
#
# ------>-----> snip ----->------>
#
# Set REQUIRE_IOC for its internal PVs
epicsEnvSet REQUIRE_IOC "REQMOD:81d1214-cslab-c-24674"
#
# Enable an exit subroutine.
dbLoadRecords "/epics/base-7.0.3.1/db/softIocExit.db" "IOC=REQMOD:81d1214-cslab-c-24674"
#
# Set E3_IOCSH_TOP for the absolute path where iocsh.bash is executed.
epicsEnvSet E3_IOCSH_TOP "/home/iocuser/e3/e3-mrfioc2/cmds"
#
# Load require module, which has the version 3.1.2
#
dlload /epics/base-7.0.3.1/require/3.1.2/lib/linux-x86_64/librequire.so
dbLoadDatabase /epics/base-7.0.3.1/require/3.1.2/dbd/require.dbd
require_registerRecordDeviceDriver
Loading module info records for require
#
# Set E3_CMD_TOP for the absolute path where emmtcaevr300.cmd exists
epicsEnvSet E3_CMD_TOP "/home/iocuser/e3/e3-mrfioc2/cmds"
#
iocshLoad 'emmtcaevr300.cmd',''
require mrfioc2,2.2.0-rc8
Module mrfioc2 version 2.2.0-rc8 found in /epics/base-7.0.3.1/require/3.1.2/siteMods/mrfioc2
    /2.2.0-rc8/
Module mrfioc2 depends on devlib2 2.9.0
Module devlib2 version 2.9.0 found in /epics/base-7.0.3.1/require/3.1.2/siteMods/devlib2/2.9.0/
Loading library /epics/base-7.0.3.1/require/3.1.2/siteMods/devlib2/2.9.0/lib/linux-x86_64/
    libdevlib2.so
Loaded devlib2 version 2.9.0
Loading dbd file /epics/base-7.0.3.1/require/3.1.2/siteMods/devlib2/2.9.0/dbd/devlib2.dbd
Calling function devlib2_registerRecordDeviceDriver
Loading module info records for devlib2
Loading library /epics/base-7.0.3.1/require/3.1.2/siteMods/mrfioc2/2.2.0-rc8/lib/linux-x86_64/
    libmrfioc2.so
Loaded mrfioc2 version 2.2.0-rc8
Loading dbd file /epics/base-7.0.3.1/require/3.1.2/siteMods/mrfioc2/2.2.0-rc8/dbd/mrfioc2.dbd
Calling function mrfioc2_registerRecordDeviceDriver
Loading module info records for mrfioc2
epicsEnvSet("IOC", "EMMTCAEVR300")
epicsEnvSet("DEV1", "EVR0")
epicsEnvSet("MainEvtCODE" "14")
epicsEnvSet("HeartBeatEvtCODE"   "122")
epicsEnvSet("ESSEvtClockRate"  "88.0525")
mrmEvrSetupPCI("EVR0",  "0a:00.0")
Notice: devPCIFindSpec() expect B:D.F in hex
Device EVR0  a:0.0 slot=11
Using IRQ 46
FWVersion 0x180e0207
```

```
Found version 519
Found SFP EEPROM
Sequencer capability detected
mTCA: Out FP:4 FPUNIV:18 RB:0 IFP:32 GPIO:2
EVR FIFO task start
Enabling interrupts
dbLoadRecords("evr-mtca-300u-ess.db","EVR=EVR0, SYS=EMMTCAEVR300, D=EVR0, FEVT=88.0525, EVNT1HZ
     =14")
var evrMrmTimeNSOverflowThreshold 100000
iocInit()
Starting iocInit
############################################################################
## EPICS R7.0.3.1-E3-7.0.3.1-patch
## EPICS Base built Jan 28 2020
############################################################################
Set EVR clock 88052500.000000
iocRun: All initialization complete
# Set the IOC Prompt String One
epicsEnvSet IOCSH_PS1 "81d1214-cslab-c-24674 > "
#
81d1214-cslab-c-24674 >
```

In addition, the PCI information is available within the running IOC via `devPCIShow` as follows:

```
81d1214-cslab-c-24674 > devPCIShow
```

Look for the line with your configuration information, in our case is:

```
PCI 0000:0a:00.0 IRQ 46
  vendor:device 10ee:7011 rev 00
```

Here the vendor id `10ee` is `Xilinx Corporation`.

The PCI information can be shown with (the second parameter is the verbosity level):

```
81d1214-cslab-c-24674 > devPCIShow 9 0x10ee
PCI 0000:0a:00.0 IRQ 46
  vendor:device 10ee:7011 rev 00
  subved:subdev 1a3e:132c
  class 118000 generic signal processing controller
  slot: 11
  driver mrf-pci
  BAR 0 32-bit MMIO    256 kB
```

4.3.4 Checking automatic configuration after reboot

Reboot and check that the module is loaded and the IOC correctly starts:

```
iocuser@cslab-ccpu-crate07: ~$ lsmod |grep mrf
mrf                    18137  0
parport                46395  1 mrf
uio                    19338  1 mrf
```

5 System in-situ verification and configuration procedure

This chapter provides the minimal system verification procedure. If it is desired to perform a more detailed testing please see Reference [see 2, p14]. It also explains how to configure the EVR in deeper detail.

Step	Goal	Info.
1	Check the EVR & EVM connection	Link status, link clock, and heartbeat timeout counter
2	Monitor receiving and acknowledging events	Event counter and receiving event frequency
3	Generate trigger signals from EVR	Various trigger signals with an oscilloscope
4	Configure hardware timestamps	Timestamps from the EVM
5	Use the EVR in standalone mode	Use the EVR without an EVM
6	Generate events from the inputs	Use the inputs
7	Share the event clock	Through the backplane clock lines

Table 5 System in-situ verification procedure

5.1 Step 1: Check the EVR and EVM connection

This assumes that the EVR is connected to a properly configured EVM. Short comments on each command or a series of commands are shown before the corresponding command.

```
#
# We can check the EVM and EVR link status and the link clock setting,
# and can also see the link counter
#
iocuser@cslab-ccpu-crate07: ~$ caget EMMTCAEVR300-EVR0:Link-Sts
EMMTCAEVR300-EVR0:Link-Sts     OK
iocuser@cslab-ccpu-crate07: ~$ caget EMMTCAEVR300-EVR0:Link-Clk-I
EMMTCAEVR300-EVR0:Link-Clk-I   88.0519
#
# Change the clock setting on the EVR, and check the link status turn to Fail
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:Link-Clk-SP 100
Old : EMMTCAEVR300-EVR0:Link-Clk-SP  88.0525
New : EMMTCAEVR300-EVR0:Link-Clk-SP  100
iocuser@cslab-ccpu-crate07: ~$ caget EMMTCAEVR300-EVR0:Link-Sts
EMMTCAEVR300-EVR0:Link-Sts     Fail
iocuser@cslab-ccpu-crate07: ~$ caget EMMTCAEVR300-EVR0:Link-Clk-I
EMMTCAEVR300-EVR0:Link-Clk-I   100
#
# Revert it back to the proper clock setting, and the link status will be OK
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:Link-Clk-SP 88.0525
Old : EMMTCAEVR300-EVR0:Link-Clk-SP  100
New : EMMTCAEVR300-EVR0:Link-Clk-SP  88.0525
iocuser@cslab-ccpu-crate07: ~$ caget EMMTCAEVR300-EVR0:Link-Sts
```

```
EMMTCAEVR300-EVR0:Link-Sts     OK
iocuser@cslab-ccpu-crate07: ~$ caget EMMTCAEVR300-EVR0:Link-Clk-I
EMMTCAEVR300-EVR0:Link-Clk-I   88.0519
#
# The link heartbeat counter is 35, depending on the time spent on Fail status
#
iocuser@cslab-ccpu-crate07: ~$ camonitor EMMTCAEVR300-EVR0:Cnt-LinkTimo-I
EMMTCAEVR300-EVR0:Cnt-LinkTimo-I 2020-02-05 14:59:19.909288 35
#
# Open another terminal to change the link clock
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:Link-Clk-SP 100
Old : EMMTCAEVR300-EVR0:Link-Clk-SP  88.0525
New : EMMTCAEVR300-EVR0:Link-Clk-SP  100
#
# In the original terminal the heartbeat counter should be increasing as follows:
#
EMMTCAEVR300-EVR0:Cnt-LinkTimo-I 2020-02-05 15:07:40.035069 36
EMMTCAEVR300-EVR0:Cnt-LinkTimo-I 2020-02-05 15:07:41.316697 37
EMMTCAEVR300-EVR0:Cnt-LinkTimo-I 2020-02-05 15:07:42.597370 38
EMMTCAEVR300-EVR0:Cnt-LinkTimo-I 2020-02-05 15:07:43.877849 39
EMMTCAEVR300-EVR0:Cnt-LinkTimo-I 2020-02-05 15:07:45.158539 40
#
# The counter will be stopped after the proper value is set again
#
# In the second terminal:
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:Link-Clk-SP 88.0525
Old : EMMTCAEVR300-EVR0:Link-Clk-SP  100
New : EMMTCAEVR300-EVR0:Link-Clk-SP  88.0525
#
# Check that the EVM is sending a timestamp
#
iocuser@cslab-ccpu-crate07: ~$ caget EMMTCAEVR300-EVR0:Time-Valid-Sts
EMMTCAEVR300-EVR0:Time-Valid-Sts Valid
```

5.2 Step 2: Monitor receiving and acknowledging events

This assumes that the EVM is sending event 14 at 14 Hz. Short comments on each command or a series of commands are shown before the corresponding command.

```
#
# Set the EVR to log event 14 with event counter E
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:EvtE-SP.OUT "@OBJ=EVR0,Code=14"
Old : EMMTCAEVR300-EVR0:EvtE-SP.OUT        @OBJ=EVR0,Code=255
New : EMMTCAEVR300-EVR0:EvtE-SP.OUT        @OBJ=EVR0,Code=14
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:EvtE-SP 14
Old : EMMTCAEVR300-EVR0:EvtE-SP            255
New : EMMTCAEVR300-EVR0:EvtE-SP            14
#
# Monitor the event counter E (14) and check the time difference between counts,
# e.g., 20616 and 20617, is 0.071429s, i.e., 14 Hz
#
iocuser@cslab-ccpu-crate07: ~$ camonitor EMMTCAEVR300-EVR0:EvtECnt-I
EMMTCAEVR300-EVR0:EvtECnt-I    2020-02-05 15:15:29.614903 20616
EMMTCAEVR300-EVR0:EvtECnt-I    2020-02-05 15:15:29.686332 20617
EMMTCAEVR300-EVR0:EvtECnt-I    2020-02-05 15:15:29.757761 20618
EMMTCAEVR300-EVR0:EvtECnt-I    2020-02-05 15:15:29.829190 20619
```

```
EMMTCAEVR300-EVR0:EvtECnt-I    2020-02-05 15:15:29.900619 20620
```

5.3 Step 3: Generate trigger signals from EVR

An EVR is composed of several logical sub-units, one of them is the pulse generator (or delay generator). Each pulse generator has an associated delay and width [2]. In the following procedure, two pulse generators are mapped to `OutFP0` and `OutFP1` of the MTCA-EVR-300U, which are connected to channels 1 and 2 of the oscilloscope respectively in order to see the effect of changing the delay and width on the output signals. This assumes that the EVM is sending event 14 at 14 Hz. Short comments on each command or a series of commands are shown before the corresponding command.

5.3.1 `OutFP0` output

```
#
# Set OutFP0 to trigger on pulse generator 0
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:OutFP0-Src-SP 0
Old : EMMTCAEVR300-EVR0:OutFP0-Src-SP 63
New : EMMTCAEVR300-EVR0:OutFP0-Src-SP 0
#
# Set pulse generator 0 to trigger on event 14
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:DlyGen0-Evt-Trig0-SP 14
Old : EMMTCAEVR300-EVR0:DlyGen0-Evt-Trig0-SP 0
New : EMMTCAEVR300-EVR0:DlyGen0-Evt-Trig0-SP 14
#
# Set the width of pulse generator 0. 10 000 is translated to 10 ms.
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:DlyGen0-Width-SP 10000
Old : EMMTCAEVR300-EVR0:DlyGen0-Width-SP 0
New : EMMTCAEVR300-EVR0:DlyGen0-Width-SP 10000
```

Figure 4 shows the output of `OutFP0` in an oscilloscope.

5.3.2 Width of pulse generator

```
#
# Change the width of pulse generator 0 from 10 ms to 50 ms
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:DlyGen0-Width-SP 50000
Old : EMMTCAEVR300-EVR0:DlyGen0-Width-SP 10000
New : EMMTCAEVR300-EVR0:DlyGen0-Width-SP 50000
```

and the output is shown in Figure 5.

It is possible to check the changes via `EMMTCAEVR300-EVR0:DlyGen0-Width-RB` as follows:

```
#
# Change the width to 50 ms, 40 ms and 80 ms in another terminal, and
# check that the Read Back (RB) value is changing
#
```

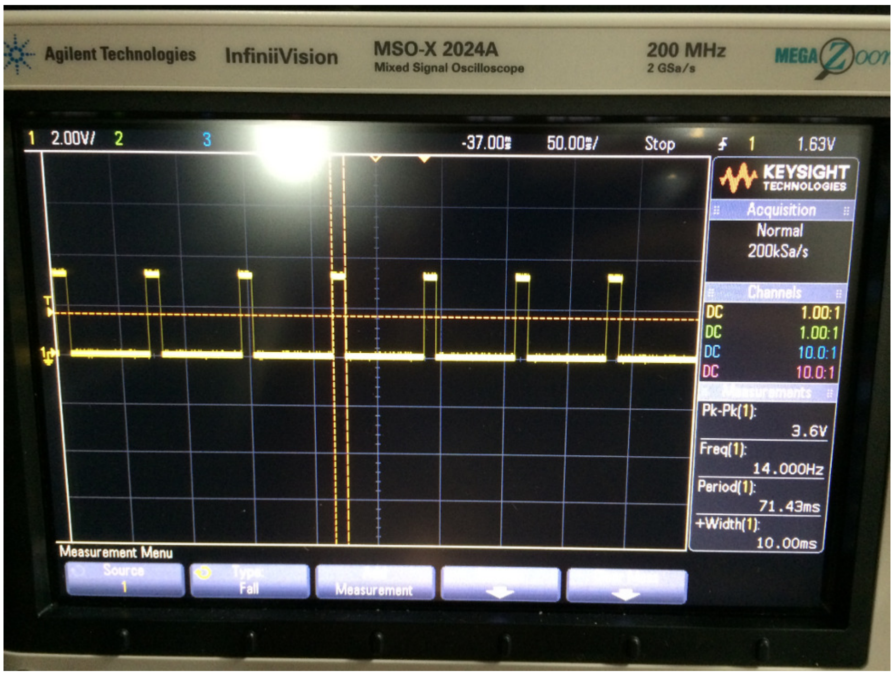

Figure 4 14 Hz signal with 10 ms width

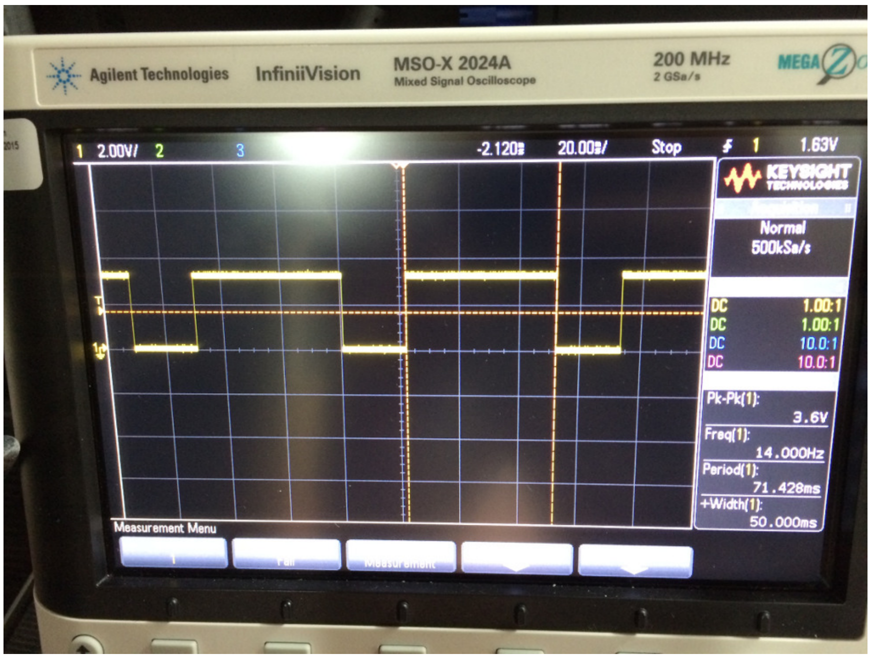

Figure 5 14 Hz signal with 50 ms width

```
iocuser@cslab-ccpu-crate07: ~$ camonitor EMMTCAEVR300-EVR0:DlyGen0-Width-RB
EMMTCAEVR300-EVR0:DlyGen0-Width-RB 2020-02-05 16:01:48.506108 50000
EMMTCAEVR300-EVR0:DlyGen0-Width-RB 2020-02-05 16:03:18.294459 40000
EMMTCAEVR300-EVR0:DlyGen0-Width-RB 2020-02-05 16:03:21.788930 80000
```

5.3.3 Delay of pulse generator

```
#
# Set the width - 20 ms - of pulse generator 0
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:DlyGen0-Width-SP 20000
Old : EMMTCAEVR300-EVR0:DlyGen0-Width-SP 80000
New : EMMTCAEVR300-EVR0:DlyGen0-Width-SP 20000
#
# Set OutFP1 to trigger on pulse generator 1
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:OutFP1-Src-SP 1
Old : EMMTCAEVR300-EVR0:OutFP1-Src-SP 63
New : EMMTCAEVR300-EVR0:OutFP1-Src-SP 1
#
# Set pulse generator 1 to trigger on event 14
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:DlyGen1-Evt-Trig0-SP 14
Old : EMMTCAEVR300-EVR0:DlyGen1-Evt-Trig0-SP 0
New : EMMTCAEVR300-EVR0:DlyGen1-Evt-Trig0-SP 14
#
# Set the width - 20 ms - of pulse generator 1
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:DlyGen1-Width-SP 20000
Old : EMMTCAEVR300-EVR0:DlyGen1-Width-SP 0
New : EMMTCAEVR300-EVR0:DlyGen1-Width-SP 20000
#
# Set the delay - 30 ms - of pulse generator 1
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:DlyGen1-Delay-SP 30000
Old : EMMTCAEVR300-EVR0:DlyGen1-Delay-SP 0
New : EMMTCAEVR300-EVR0:DlyGen1-Delay-SP 30000
```

Figure 6 shows the result.

The configuration of the outputs and pulse generators can be added to the startup script after `iocInit()` replacing `caput` by `dbpf`.

5.4 Step 4: Configure hardware timestamps

This section provides the minimal information to configure an IOC to get its timestamps from the timing system. More information about timestamping can be found in the document ESS-0085848 (ICS Engineering Manual for Timestamping). Short comments on each command or a series of commands are shown before the corresponding command.

```
#
# Set up the Time-I record to process on arrival of event 14
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:Time-I.EVNT 14
Old : EMMTCAEVR300-EVR0:Time-I.EVNT  125
New : EMMTCAEVR300-EVR0:Time-I.EVNT  14
#
```

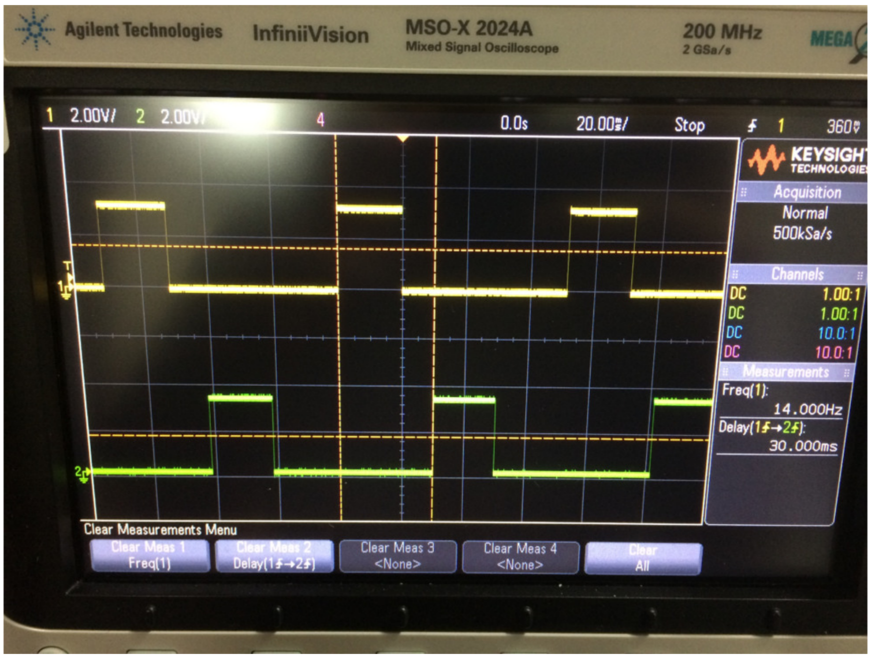

Figure 6 Two 14 Hz signals with 30 ms delay

```
# Set up the Time-I record to use the hardware timestamp of event 14
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:Time-I.INP "@OBJ=EVR0, Code=14"
Old : EMMTCAEVR300-EVR0:Time-I.INP   @OBJ=EVR0, Code=125
New : EMMTCAEVR300-EVR0:Time-I.INP   @OBJ=EVR0, Code=14
#
# Set up your records to use the timestamp from the Time-I record
#
iocuser@cslab-ccpu-crate07: ~$ caput examplerecord.TSEL EMMTCAEVR300-EVR0:Time-I.TIME
Old : examplerecord.TSEL
New : examplerecord.TSEL             EMMTCAEVR300-EVR0:Time-I.TIME NPP NMS
#
# Check that your records have the same timestamp as event 14
#
iocuser@cslab-ccpu-crate07: ~$ caget -a examplerecord EMMTCAEVR300-EVR0:EvtECnt-I
examplerecord                  2019-01-04 11:21:17.183434 1171
EMMTCAEVR300-EVR0:EvtECnt-I    2019-01-04 11:21:17.183434 3385
```

5.4.1 Configure the exact tick period

The timing system stores internally the timestamps as ticks, that then EPICS translates to wall-clock time. To do this the IOC should be configured with the actual tick period. By default it assumes that the event link frequency is being generated by the EVM fractional synthesizer. If the EVM takes the event frequency from an external source through its inputs, the exact frequency has to be set in the EVR IOC.

```
#
# Set the exact event frequency
```

```
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:Time-Clock-SP 88.0525
Old : EMMTCAEVR300-EVR0:Time-Clock-SP 0
New : EMMTCAEVR300-EVR0:Time-Clock-SP 88.0525
```

5.5 Step 5: Use the EVR in standalone mode

The EVR can work on its own without a connection to an EVM, but with reduced functionality. Here it is shown how to configure the EVR to work in this way. Some of the settings need to be included in the startup script shown in Listing 4.2, after `iocInit()`, so it is advised to include all the configuration in that file. Short comments on each command or a series of commands are shown before the corresponding command.

```
### Get current time from system clock, this will be used for the timestamps ###
dbpf $(IOC)-$(DEV1):TimeSrc-Sel "Sys. Clock"

### Set up the prescaler that will trigger the sequencer at 14 Hz ###
# The value of the prescaler is the integer which gives the expected frequency (14 Hz in this
    example) when the event frequency (88.0525 MHz for ESS) is divided by the integer: 88.0525
    MHz / 6289464 = 14 Hz
dbpf $(IOC)-$(DEV1):PS0-Div-SP 6289464

### Set up the sequencer ###
# Set the runmode to normal, so that the sequencer re-arms after it finishes running
dbpf $(IOC)-$(DEV1):SoftSeq0-RunMode-Sel "Normal"
# Set the trigger of the sequencer as prescaler 0
dbpf $(IOC)-$(DEV1):SoftSeq0-TrigSrc-2-Sel "Prescaler 0"
# Set the engineering units (microseconds) for the delay of the events in the sequence (sequence
    timestamps) used in configure_sequencer_14Hz.sh; more information below
dbpf $(IOC)-$(DEV1):SoftSeq0-TsResolution-Sel "uSec"
# Attach the soft sequence to a specific hardware sequence
dbpf $(IOC)-$(DEV1):SoftSeq0-Load-Cmd 1
# Enable the sequencer
dbpf $(IOC)-$(DEV1):SoftSeq0-Enable-Cmd 1

### Run the script that configures the events and timestamps of the sequence, more information
    below ###
system("/bin/sh ./configure_sequencer_14Hz.sh $(IOC) $(DEV1)")
```

The file `configure_sequencer_14Hz.sh` shown in Listing 5.1 should be located in the same directory as `emmtcaevr300.cmd`; it specifies what events and with what delay after the sequencer is triggered (what is known as sequencer timestamps) should be sent:

```
### Bash script to configure the EVM/EVR sequencer
### All values in us, as configured with $1-$2:SoftSeq0-TsResolution-Sel

### Set up the sequence content, events and timestamps
### Event 127 is always needed at the end, it is the end-of-sequence event and stops the sequencer
### The first event in the its list is sent with the first delay in its list, the second event
    after the second delay (the start of time is always the moment when the sequencer is
    triggered) and so on
### The timestamps should be monotonically increasing
# Event code 14 (14 Hz), 127 is the end of sequence
caput -a $1-$2:SoftSeq0-EvtCode-SP 2 14 127
# Defining time at which the event codes are sent in us (timestamps), as configured with $1-$2:
    SoftSeq0-TsResolution-Sel
caput -a $1-$2:SoftSeq0-Timestamp-SP 2 0 1
```

```
# Commit the sequence to HW
caput $1-$2:SoftSeq0-Commit-Cmd 1
```

Listing 5.1 Sequencer population file `configure_sequencer_14Hz.sh`.

For more information, please check ESS-0331569 (ICS Engineering Manual for MRF mTCA-EVM-300), the EVR sequencer acts in exactly the same way as the EVM sequencer, but the EVR only has one sequencer.

5.6 Step 6: Generate events from the inputs

The MTCA-EVR-300(U) has two inputs that can be used to trigger events, which can be timestamped or used to trigger processing or sequencers. Short comments on each command or a series of commands are shown before the corresponding command.

```
#
# Set FPIN0 to generate an event on the rising edge of the input signal
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:In0-Trig-Ext-Sel "Edge"
Old : EMMTCAEVR300-EVR0:In0-Trig-Ext-Sel Off
New : EMMTCAEVR300-EVR0:In0-Trig-Ext-Sel Edge
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:In0-Edge-Sel "Active Rising"
Old : EMMTCAEVR300-EVR0:In0-Edge-Sel Active Rising
New : EMMTCAEVR300-EVR0:In0-Edge-Sel Active Rising
#
# Select the event number to be generated
#
iocuser@cslab-ccpu-crate07: ~$ caput EMMTCAEVR300-EVR0:In0-Code-Ext-SP 10
Old : EMMTCAEVR300-EVR0:In0-Code-Ext-SP 0
New : EMMTCAEVR300-EVR0:In0-Code-Ext-SP 10
```

5.7 Step 7: Share the event clock

The MTCA-EVR-300(U) can share the event clock with the rest of the AMCs in the μTCA crate. This process is explained in the document ESS-0508492 (ICS Engineering Manual for μTCA Backplane Clock Distribution).

6 Troubleshooting

This chapter shows some errors that can happen when installing and configuring the MRF MTCA-EVR-300(U).

6.1 PCI error

If the device file permission is wrong, one can see the following error:

```
mrmEvrSetupPCI("EVR0",  "0b:00.0")
Notice: devPCIFindSpec() expect B:D.F in hex
Device EVR0  b:0.0 slot=6
```

```
Using IRQ 16
Can neither open resource file nor uio file of PCI device 0000:0b:00.0 BAR 0
PCI error: Failed to map BARs 0 for EC 30
```

Glossary

Term	Definition
CPU	Central Processing Unit
E3	ESS EPICS environment
EPICS	Experimental Physics and Industrial Control System
ESS	European Spallation Source
EVM	Event Master
EVR	Event Receiver
FPGA	Field-Programmable Gate Array
ICS	Integrated Control System
I/O	Input/Output
IOC	Input/Output Controller
MCH	MTCA Carrier Hub
MRF	MicroResearch Finland
MTCA	Micro Telecommunications Computing Architecture
RTM	Rear Transition Module
SFP	Small Form-factor Pluggable

Bibliography

[1] MRF Technical Reference. *Event System with Delay Compensation Technical Reference Firmware 0205*, April 26, 2016.

[2] Michael Davidsaver. *EVR User Guide*, August, 2015. URL http://epics.sourceforge.net/mrfioc2/evr-usage.pdf.

Document revision history

Revision	**Reason for and description of change**	**Author**	**Date**
1	First release	Javier Cereijo Garcia	February 11, 2020